Wie beweisen Computer Theoreme und was bedeutet das für uns?

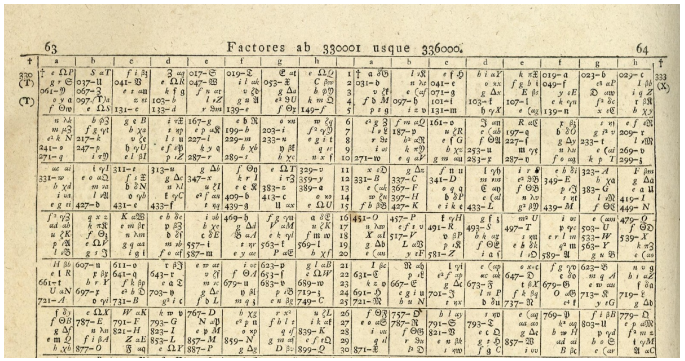
Nima Rasekh

Universität Greifswald

18.12.2024

Was ist ein Computer/Rechner?

1. In 1797 oder sogar 1950 wäre es eine Person:

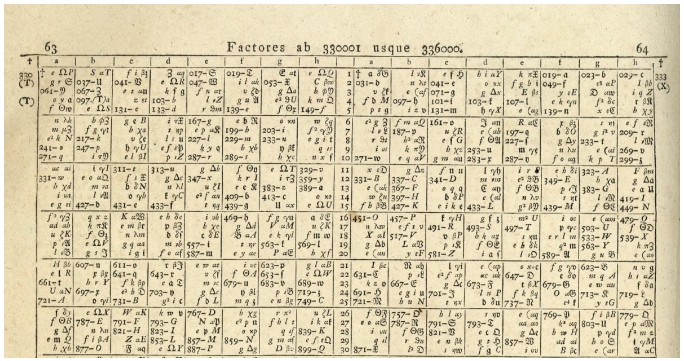


(a) Primzahltabelle von Felkel–Vega (1797)

(b) Menschliche Computer (1950)

Was ist ein Computer/Rechner?

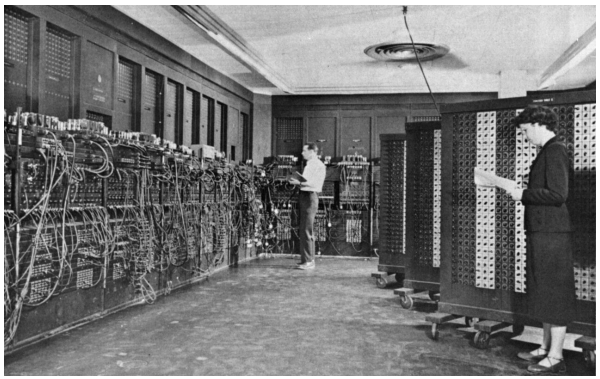
1. In 1797 oder sogar 1950 wäre es eine Person:



(a) Primzahltabelle von Felkel–Vega (1797)

(b) Menschliche Computer (1950)

2. Ab den 40er Jahren wird daraus langsam ein elektronisches Gerät

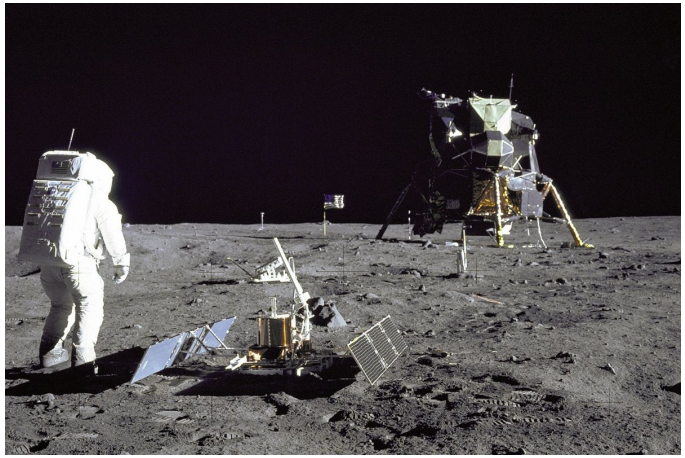


(a) ENIAC (1945)

(b) IBM Blue Gene (2006)

Computer in der Wissenschaft

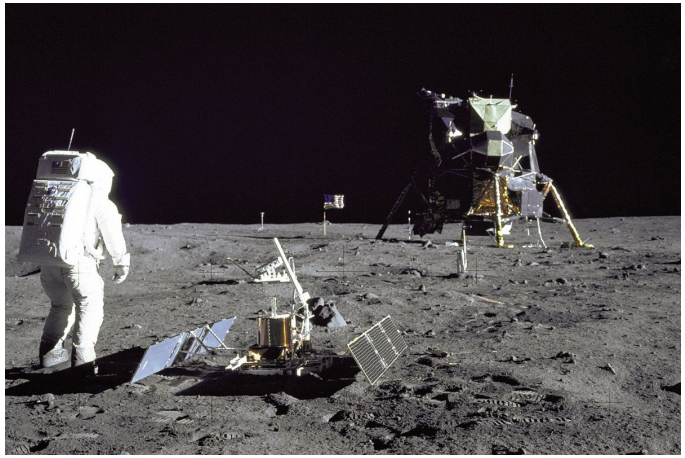
(a) Harvard Computers (1890)



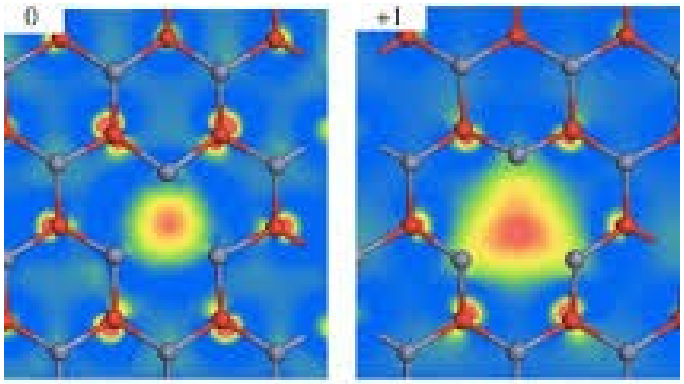
(b) Apollo Guidance Computer (1969)

Computer in der Wissenschaft

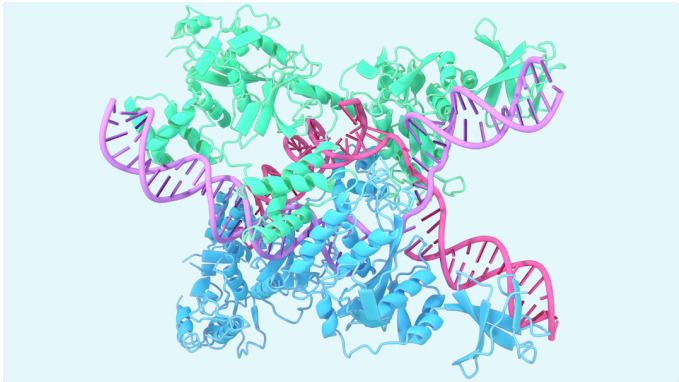
(a) Harvard Computers (1890)



(b) Apollo Guidance Computer (1969)

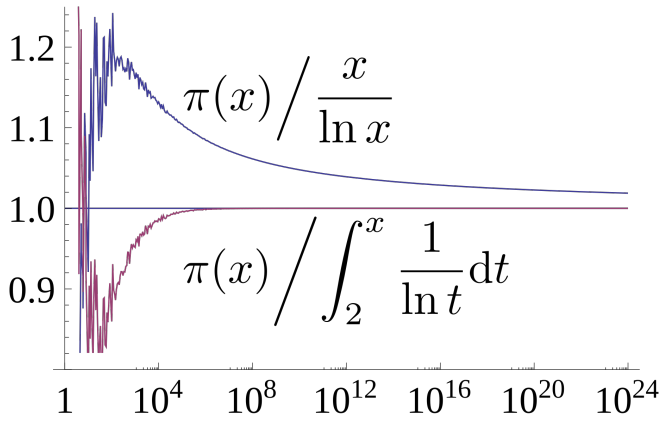


(c) CASTEP (1990)

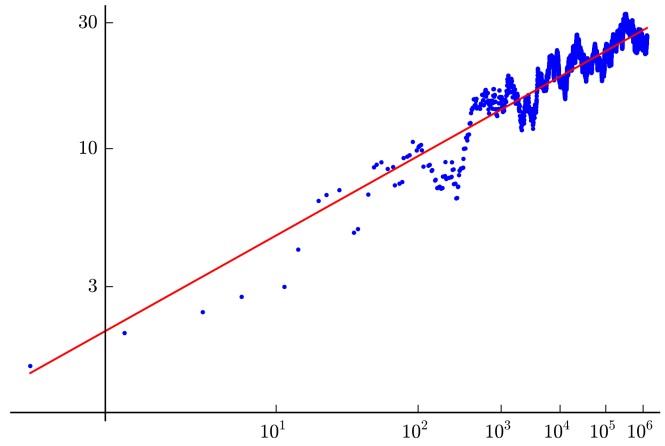


(d) AlphaFold (2018)

Computer in der Mathematik

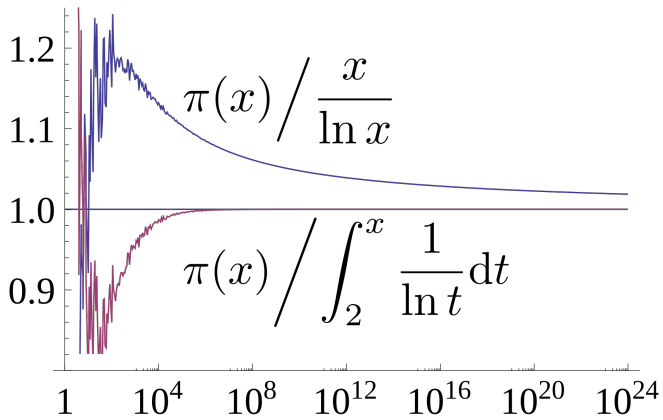


(a) Primzahlsatz (1798)

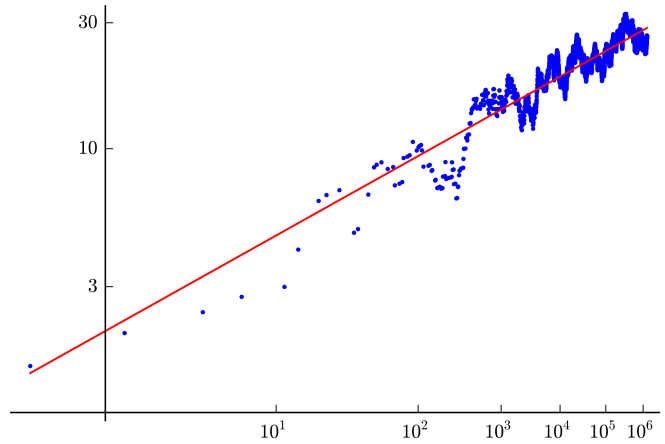


(b) Vermutung von Birch und Swinnerton-Dyer (1965)

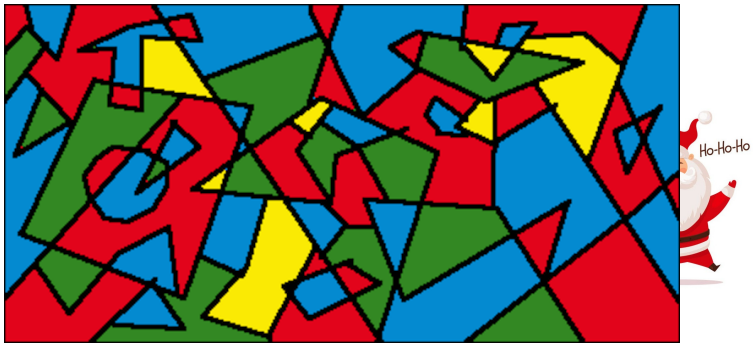
Computer in der Mathematik



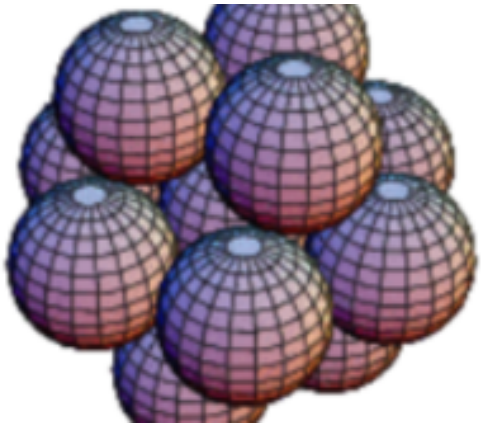
(a) Primzahlsatz (1798)



(b) Vermutung von Birch und Swinnerton-Dyer (1965)



(c) Vier-Farben-Satz (1967)



(d) Keplervermutung (1998)

Da fehlt etwas: Beweise!

Computer haben viele Anwendungen gefunden:

1. Berechnungen
2. Vermutungen
3. Beweise durch Überprüfung von endlich vielen Fällen

Da fehlt etwas: Beweise!

Computer haben viele Anwendungen gefunden:

1. Berechnungen
2. Vermutungen
3. Beweise durch Überprüfung von endlich vielen Fällen

Was fehlt?

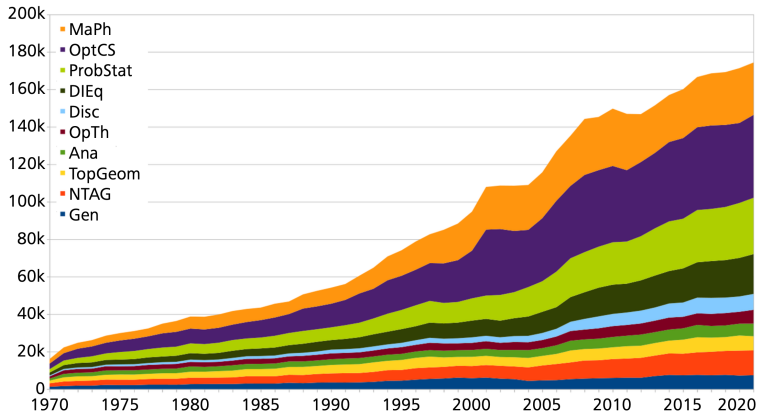
- Beweise!

Fragen:

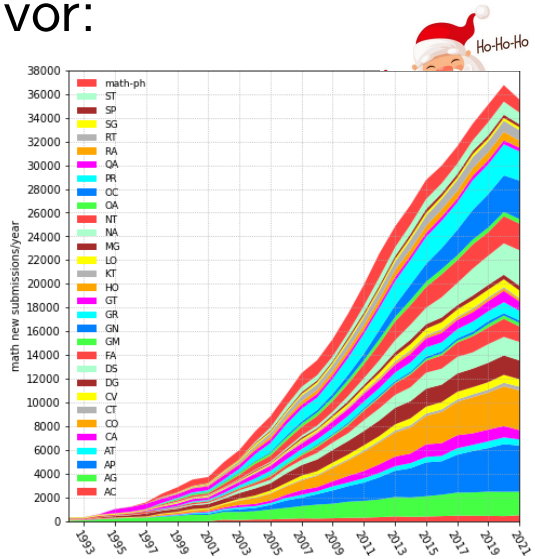
1. Können Computer Theoreme beweisen und wie?
2. Können Computer Beweise automatisieren?
3. Können Computer neue Theoreme beweisen?

Warum sollen Computer etwas beweisen?

1. Es gibt mehr Mathematik als je zuvor:



Aktiv publizierende Personen 1970 - 2020
Klaus Hulek, Olaf Teschke. How do mathematicians publish? EMS Mag. 129



Anzahl der Papiere in Math auf arXiv
https://info.arxiv.org/help/stats/2021_by_area/index.html

Verlieren die Übersicht, aber ist besser für ML Algorithmen!

2. Computer Berechnungen können nicht kontrolliert werden.
3. Mathematik ist komplizierter: Fehler sind wahrscheinlicher.

2 Beispiele

1. **Homotopie Hypothese:** Voevodsky[1]
   - Publizierter Beweis von Voevodsky und Kapranov in 1991
   - Gegenbeispiel von Simpson in 1998
   - Kein konkreter Fehler im Papier entdeckt bis 2013

"A technical argument by a trusted author, which is hard to check and looks similar to arguments known to be correct, is hardly ever checked in detail."

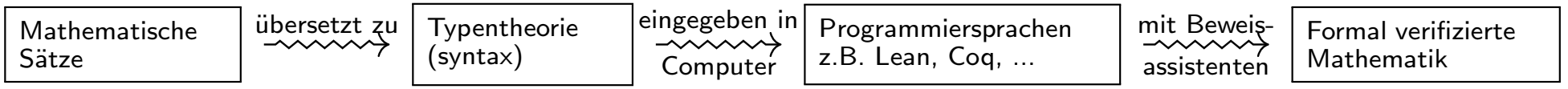
2. **Verdichtete Mathematik:** Scholze[2]
   - Komplizierter Beweis, schwer zu überprüfen
   - Hat das *Liquid Tensor Experiment* vorgeschlagen

"I learnt that it can now be possible to take a research paper and just start to explain lemma after lemma to a proof assistant, until you've formalized it all! I think this is a landmark achievement."

[1] https://www.ias.edu/ideas/2014/voevodsky-origins

[2] https://xenaproject.wordpress.com/2020/12/05/liquid-tensor-experiment/

Wie kann ein Computer etwas beweisen?

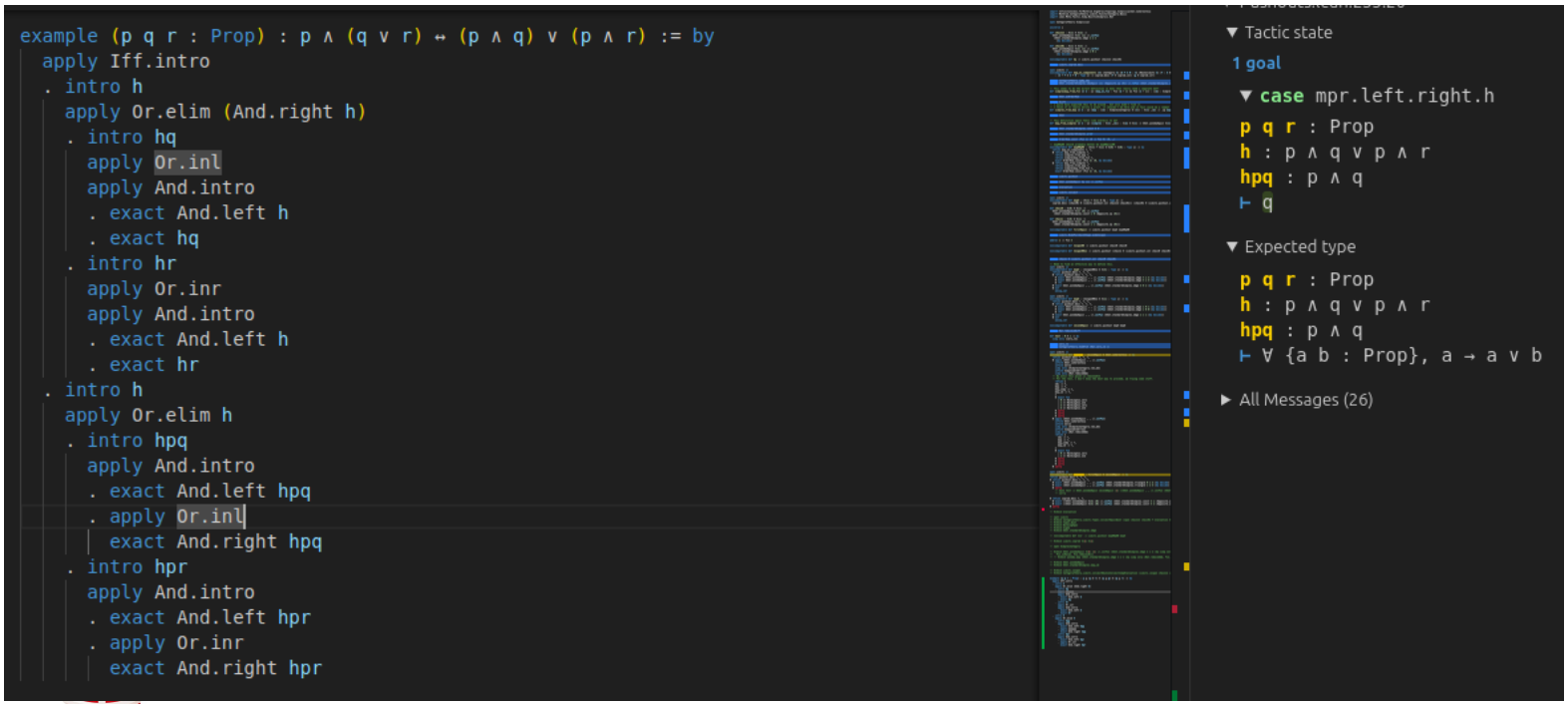


Wie kann ein Computer etwas beweisen?

Mathematische Sätze → übersetzt zu → Typentheorie (syntax) → eingegeben in Computer → Programmiersprachen z.B. Lean, Coq, ... → mit Beweisassistenten → Formal verifizierte Mathematik

Beispiel

Für Aussagen p, q, r gilt: $p \wedge (q \vee r) \leftrightarrow (p \wedge q) \vee (p \wedge r)$



Aber gibt es Formalisierungen echter Mathematik?

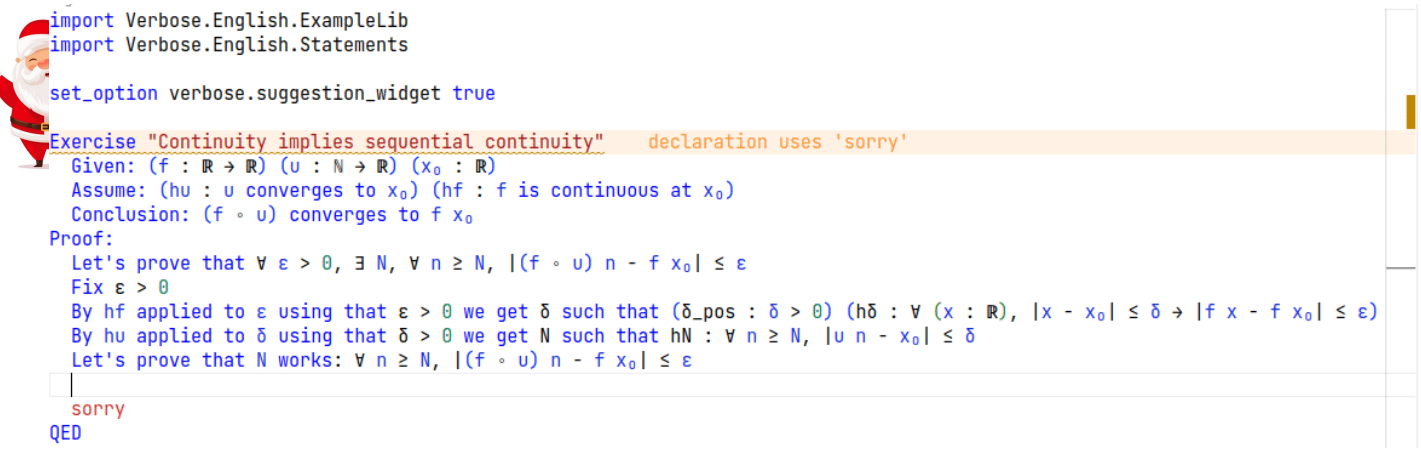
Hier sind paar aktuelle Entwicklungen:

1. **Zahlentheorie:** Großer Fermatscher Satz, seit 2024[3]
2. **Analysis:** Satz von Carleson, seit 2024[4]
3. **Algebra:** Liquid Tensor Experiment, 2020 - 2022[5]
4. **Differentialgeometrie:** Umstülpung der Sphäre, 2020 - 2022[6]
5. **Topologie:** Homotopiegruppe $\pi_4(S^3)$, 2016 - 2022[7]
6. **Geometrie:** Keplervermutung, 2003 - 2015[8]

3 https://github.com/ImperialCollegeLondon/FLT
4 https://github.com/fpvandoorn/carleson
5 https://github.com/leanprover-community/lean-liquid
6 https://github.com/leanprover-community/sphere-eversion
7 https://github.com/agda/cubical/tree/master/Cubical/Homotopy/Group/Pi4S3
8 https://github.com/flyspeck/flyspeck

Wohin kann es führen?

1. Journal Einreichung mit Formalisierung[9]
2. Automatische Beweise via KI[10]
3. Anwendung in der Lehre[11,12,13,14,15]



```
import Verbose.English.ExampleLib
import Verbose.English.Statements

set_option verbose.suggestion_widget true

Exercise "Continuity implies sequential continuity"    declaration uses 'sorry'
  Given: (f : ℝ → ℝ) (u : ℕ → ℝ) (x₀ : ℝ)
  Assume: (hu : u converges to x₀) (hf : f is continuous at x₀)
  Conclusion: (f ∘ u) converges to f x₀
Proof:
  Let's prove that ∀ ε > 0, ∃ N, ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε
  Fix ε > 0
  By hf applied to ε using that ε > 0 we get δ such that (δ_pos : δ > 0) (hδ : ∀ (x : ℝ), |x - x₀| ≤ δ → |f x - f x₀| ≤ ε)
  By hu applied to δ using that δ > 0 we get N such that hN : ∀ n ≥ N, |u n - x₀| ≤ δ
  Let's prove that N works: ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε

  sorry
QED
```

▼ Tactic state

1 goal

```
f : ℝ → ℝ
u : ℕ → ℝ
x₀ : ℝ
hu : u converges to x₀
hf : f is continuous at x₀
ε : ℝ
ε_pos : ε > 0
δ : ℝ
δ_pos : δ > 0
hδ : ∀ (x : ℝ), |x - x₀| ≤ δ → |f x - f x₀| ≤ ε
N : ℕ
hN : ∀ n ≥ N, |u n - x₀| ≤ δ
⊢ ∀ n ≥ N, |(f ∘ u) n - f x₀| ≤ ε
```

▼ Suggestions

Use shift-click to select sub-expressions.

9 https://xenaproject.wordpress.com/2023/11/04/formalising-modern-research-mathematics-in-real-time/

10 https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/

11 https://impermeable.github.io/

12 https://gihanmarasingha.github.io/modern-maths-pages/

13 https://cs22.io/

14 https://hhu-adam.github.io/

15 https://github.com/PatrickMassot/lean-verbose

Was bedeutet das für mich?

Hier sind ein paar erste mögliche Schritte:

1. Auf dem Laufendem zu bleiben.
   - Meilensteine verfolgen: LTE[16], AlphaProof[17], ...
   - Formalisierung an (deutschen) Unis: Bonn[18], Düsseldorf[19]
2. Ein bisschen Lean versuchen.[20]
3. Paar erste Definitionen und Ergebnisse des eigenen Forschungsbereichs formalisieren.
4. Formalisierung in der Lehre integrieren (z.B. Übungen).
5. Formalisierung eines wichtigen Beweises.

Falls es Fragen oder Interesse gibt, gerne nachfragen!

16 https://leanprover-community.github.io/blog/posts/lte-final/
17 https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/
18 https://florisvandoorn.com/
19 https://hhu-adam.github.io/
20 https://adam.math.hhu.de/#/g/leanprover-community/nng4

Formalisierung in Lean

Wie viele 

?

Demonstration